





## Article

# A Fast Matrix Compression Method for Large Scale Numerical Modelling of Rotationally Symmetric 3D Passive Structures in Fusion Devices

Francesca Cau <sup>1</sup>, Andrea Gaetano Chiariello <sup>2</sup>, Guglielmo Rubinacci <sup>3</sup>, Valentino Scalera <sup>4</sup>,  
Antonello Tamburrino <sup>5</sup>, Salvatore Ventre <sup>5,\*</sup> and Fabio Villone <sup>3</sup>

- <sup>1</sup> Fusion for Energy (F4E), Carrer de Josep Pla, 9, 08019 Barcelona, Spain; francesca.cau@f4e.europa.eu  
<sup>2</sup> CREATE, DI, Università degli Studi della Campania “Luigi Vanvitelli”, Via Roma, 28, 80125 Aversa, Italy; andrea.gaetano.chiariello@unicampania.it  
<sup>3</sup> CREATE, DIETI, Università degli Studi di Napoli Federico II, Via Claudio, 21, 80125 Napoli, Italy; rubinacci@unina.it (G.R.); fabio.villone@unina.it (F.V.)  
<sup>4</sup> Engineering Department, University of Naples Parthenope, 80143 Napoli, Italy; valentino.scalera@uniparthenope.it  
<sup>5</sup> CREATE, DAEIMI, Università degli Studi di Cassino e del Lazio Meridionale, Via G. Di Biasio, 43, 03043 Cassino, Italy; tamburrino@unicas.it  
\* Correspondence: ventre@unicas.it



**Citation:** Cau, F.; Chiariello, A.G.; Rubinacci, G.; Scalera, V.; Tamburrino, A.; Ventre, S.; Villone, F. A Fast Matrix Compression Method for Large Scale Numerical Modelling of Rotationally Symmetric 3D Passive Structures in Fusion Devices. *Energies* **2022**, *15*, 3214. <https://doi.org/10.3390/en15093214>

Academic Editor: Hiroshi Sekimoto

Received: 25 March 2022

Accepted: 25 April 2022

Published: 27 April 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Abstract:** This paper illustrates the development of a recursive QR technique for the analysis of transient events, such as disruptions or scenario evolution, in fusion devices with three-dimensional conducting structures using an integral eddy current formulation. An integral formulation involves the solution, at each time step, of a large full linear system. For this reason, a direct solution is impractical in terms of time and memory consumption. Moreover, typical fusion devices show a symmetric/periodic structure. This can be properly exploited when the plasma and other sources possess the same symmetry/periodicity of the structure. Indeed, in this case, the computation can be reduced to only a single sector of the overall structure. In this work the periodicity and the symmetries are merged in the recursive QR technique, exhibiting a huge decrease in the computational cost. Finally, the proposed technique is applied to a realistic large-scale problem related to the International Thermonuclear Experimental Reactor (ITER).

**Keywords:** plasma fusion; integral formulations; fast methods; eddy current; QR-MGS sparsification; H-matrix

## 1. Introduction

The working principle of magnetic confinement fusion devices is fundamentally based on the electromagnetic interaction of the plasma, where fusion reactions occur with the conducting structures surrounding the plasma, and where external currents circulate. Such conductors can be both actively fed (e.g., poloidal field, PF, coils and toroidal field, TF, and coils) and passive (e.g., the vacuum vessel). The currents flowing in the former set are responsible for the magnetic field needed to keep the plasma in equilibrium, hence giving rise to the nominal desired plasma configuration. Conversely, passive conductors play a fundamental role in the stability of such an equilibrium configuration. In fact, assuming the plasma is described as a current-carrying fluid using the Magneto-Hydro-Dynamics (MHD) model, it can be demonstrated that there exist unstable modes of evolution—the so-called MHD instabilities. A huge number of such unstable modes may exist, whose triggering depends on several physical and geometrical plasma parameters, such as pressure, current, position, and shape. They are usually classified in terms of the toroidal and poloidal number, i.e., the Fourier decomposition in the toroidal and poloidal direction of the dominant plasma perturbation. In the present paper, we consider situations in which the plasma

evolves in an axisymmetric way, although this is not an intrinsic limitation of the proposed approach, as we will clarify below. Practically, this means that the plasma geometrical descriptors (e.g., the centroid vertical position, gaps with respect to the first wall, and the displacement with respect to the nominal configuration) may grow exponentially in time, after an initial perturbation takes place. The time scale of such phenomenon—the so-called growth rate of the instability—may be as fast as microseconds in typical fusion devices, in the absence of passive conducting structures, being related to the inertial dynamics of the plasma, which is very fast due to its very small mass. Such abrupt movements and deformations of the current-carrying plasma induce eddy currents in surrounding passive conductors; such eddy currents, in turn, tend to counteract the plasma movement, thanks to the Lenz rule, hence slowing down the instability to the time scale over which eddy currents decay—tens or hundreds of milliseconds for typical devices. This allows an active control of the instability to be practically feasible [1].

The discussion reported above demonstrates clearly that numerical modelling of the conducting structures surrounding the plasma is of paramount importance for a macroscopic description of the behaviour of a fusion device. A comprehensive, modern, and rigorous presentation of the theory leading to the numerical modelling of such complex electromagnetic problems can be found in the book of Bossavit [2]. In its exposition, Bossavit limits the attention to the variational formulation of the field problems. In this case, the quasi-stationary approximation of the Maxwell equations are discretized by the Finite Element Method (FEM). Here, the computational domain in principle also includes the air. This approach is very well diffused (see for instance [3]) and it is at the basis of many general-purpose commercial codes. For avoiding the discretization of the air, a boundary integral equation can be coupled to the differential equations on the boundary of the conducting domain, leading to the so-called FEM-BEM (Boundary Element Method), as described for instance in [2,4,5]. The thermonuclear fusion devices present very complex geometries in which massive structures and thin shells coexist and ferromagnetic effects are usually negligible. In this respect, integral formulations are usually quite advantageous, because they allow (i) to mesh the conductors only and (ii) to implicitly take into account regularity conditions at infinity. For this reason, in the first generation of tokamaks, made almost completely of thin conducting structures, the finite element eddy current integral formulation for nonmagnetic shells proposed by Kameari [6], Bossavit [7], and Blum et al. [8] resulted in being particularly efficient, from the computational point of view. The main obstacle to the treatment of fully three-dimensional massive conductors was the lack of a general method for generating a complete set of independent, solenoidal shape functions for the current density. The proposal [9] of a numerical formulation for modelling in an efficient way these 3D massive structures was the main step leading to the implementation of the CARIDDI code [9,10], and indeed it has been extensively used for the modelling of fusion devices, such as for instance JET, EAST, COMPASS, RFX, JT-60SA, ITER, and DEMO. On the other hand, general purpose commercial codes based on differential formulations substantially improved in the course of the years [11–14]. Nowadays, they represent a valid alternative in many cases where the presence of the plasma does not pose specific challenges in the modelling.

One drawback of integral formulations—and of course CARIDDI is no exception—is that they give rise to fully populated matrices to be inverted to find the solution. This inevitably poses a limitation to the maximum number of discrete unknowns—the Degrees Of Freedom, DOF—that can be practically handled. On the other hand, the complexity of devices and the accuracy required for the analyses require very detailed models and hence huge numerical models to be considered. To tackle this problem, so-called “fast methods” can be used, ranging from FFT (Fast Fourier Transform)-based approaches [15–17] to multipole approximations [18,19], to QR-recursive compression [20,21]. Although all such methods have been successfully implemented in the CARIDDI code, our previous experience clearly show that for fusion devices, the most effective technique is the QR-recursive (see [22] for the definition of QR matrix factorization). This is the reason why, in the present

paper, we focus on a number of significant extensions of such a technique, which computes the effective numerical solution of the problem with a very high efficiency, hence allowing an unprecedented level of detail in the description of fusion devices.

In the present paper, the authors starting from the previous version of the QR-recursive method, face some important numerical issues, adding new efficient enhancements. We can summarize the new features addressed by the paper in the following:

- The extension of the method when both the structure and the sources exhibit the same symmetries. This is quite important because symmetry is typical in devices for fusion applications.
- A new approach (namely the DOF-based method) is introduced for the QR-recursive method, which compared to the old one (ELEMENT-based method) is numerically more effective.
- An efficient numerical approach is used in solving the system in order to handle the “electrodes” case. This case is very usual for practical fusion device application.
- We tackle the problem of the “small boxes” in QR-recursive, which could degrade the performances of the overall method.

Finally, we applied the method to a relevant application case of the International Thermonuclear Experimental Reactor (ITER) [23].

## 2. Mathematical Formulation

### 2.1. Integral Formulation

Here we summarize the Magneto-Quasi-Static volume integral formulation at the basis of the numerical model. We refer to a conducting three-dimensional domain  $V_c$  excited by a time varying magnetic field and by a set of current/voltage sources applied at a subset  $S_E$  of its boundary made by a set of equipotential electrodes.

Faraday’s law is automatically satisfied by assuming

$$\mathbf{E} = -\frac{\partial \mathbf{A}}{\partial t} - \nabla \varphi, \quad \text{in } \mathbb{R}^3. \quad (1)$$

The magnetic vector potential is uniquely defined by  $\nabla \times \mathbf{A} = \mathbf{B}$  with the Coulomb gauge and the regularity conditions at infinity. Consequently, it is possible to express  $\mathbf{A}$  in terms of the unknown solenoidal current density, by using the Biot–Savart law:

$$\mathbf{A}(\mathbf{r}, t) = \frac{\mu_0}{4\pi} \int_{V_c} \frac{\mathbf{J}(\mathbf{r}', t)}{|\mathbf{r} - \mathbf{r}'|} dV' + \mathbf{A}_s(\mathbf{r}, t), \quad \text{in } \mathbb{R}^3, \quad (2)$$

where

$$\mathbf{A}_s(\mathbf{r}, t) = \frac{\mu_0}{4\pi} \int_{\mathbb{R}^3 - V_c} \frac{\mathbf{J}_s(\mathbf{r}', t)}{|\mathbf{r} - \mathbf{r}'|} dV', \quad \text{in } \mathbb{R}^3, \quad (3)$$

and  $\mathbf{J}_s$  is the (known) current density due to the external sources.

The constitutive equation

$$\mathbf{J} = \sigma \mathbf{E}, \quad \text{in } V_c, \quad (4)$$

where  $\sigma$  is the electric conductivity tensor, can be verified in an average sense by adopting a weighted residual approach. It results in the following weak form

$$\int_{V_c} \mathbf{w} \cdot \left\{ \sigma^{-1} \mathbf{J} + \frac{\partial}{\partial t} \left[ \frac{\mu_0}{4\pi} \int_{V_c} \frac{\mathbf{J}(\mathbf{r}', t)}{|\mathbf{r} - \mathbf{r}'|} dV' \right] + \frac{\partial \mathbf{A}_s}{\partial t} \right\} dV + \sum_{h=1, N_E} \int_{S_E} \Phi_h \mathbf{w} \cdot \hat{\mathbf{n}} dS = 0 \quad (5)$$

for any  $\mathbf{w} \in S$  and with  $\mathbf{J} \in S$ . Here  $S$  is the set of solenoidal vector functions with zero normal component on  $\partial V_c \setminus S_E$ , being  $\hat{\mathbf{n}}$  the unit normal pointing outward  $V_c$ :

$$S = \left\{ \mathbf{w} \in H(\text{div}; V_c) \mid \text{div } \mathbf{w} = 0 \text{ in } V_c \text{ and } \mathbf{w} \cdot \hat{\mathbf{n}} = 0 \text{ on } \partial V_c \setminus S_E \right\}. \quad (6)$$

$N_E$  is the number of electrodes (they are part of the boundary of the conducting domain), identified by the surface  $S_E$ . At each electrode  $h$ , there is an unknown potential  $\Phi_h$  and a prescribed current  $I_h(t)$ .

In Figure 1 we report a schematic sketch showing the geometrical objects involved in Equation (5). Note that, being an integral formulation, only the sources in the conducting domain are involved in the computation of the integrals. The air regions do not bring any additional unknowns.

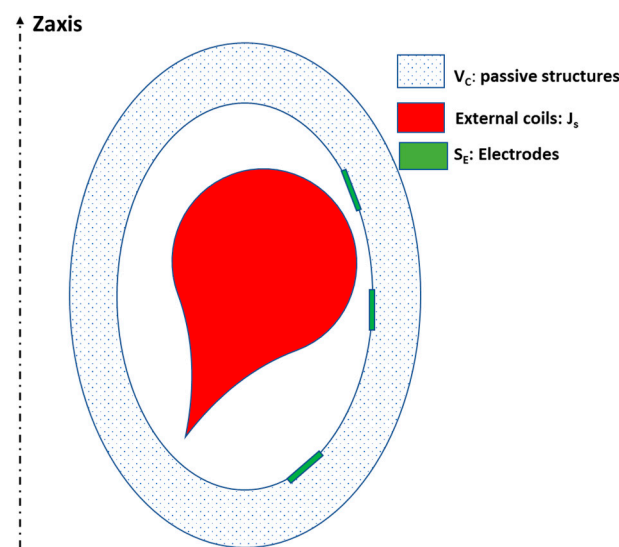


Figure 1. A schematic figure reporting the meaning of the symbols used in Equation (5).

### 2.2. Numerical Model

A numerical solution is obtained by applying Galerkin’s method to Equation (5). The unknown is the current density vector  $\mathbf{J}$ , which we represent as the linear combination of  $N$  basis functions  $\mathbf{J}_i \in S$ :

$$\mathbf{J}(\mathbf{r}, t) = \sum_{i=1:N} I_i(t) \mathbf{J}_i(\mathbf{r}). \quad (7)$$

According to the Galerkin’s method, we choose  $N$  independent weighting functions as  $\mathbf{W}_i = \mathbf{J}_i$ . Condition  $\mathbf{J}_i \in S$  can be satisfied by introducing the electric vector potential  $\mathbf{T}$  ( $\mathbf{J} = \nabla \times \mathbf{T}$ ) and adopting edge element shape functions for  $\mathbf{T}$ . The uniqueness of the vector potential is assured by the tree-cotree gauge [10]. This gauge is conveniently imposed directly on the basis functions, introducing the tree-cotree decomposition of the mesh and eliminating the degrees of freedom associated to the tree edges. Condition  $\mathbf{J}_i \cdot \hat{\mathbf{n}} = 0$  on  $\partial V_c \setminus S_E$  can also be imposed on the shape functions using the approach described in [10,24,25]. The shape functions  $\mathbf{J}_i$  are therefore derived from the  $N_i$  edge element functions for the gauged vector potential:

$$\mathbf{J}_i = \nabla \times \mathbf{N}_i. \quad (8)$$

In this way, Galerkin’s method applied to (5) gives the following linear dynamical system, for  $t \geq 0$ :

$$\mathbf{L} \frac{d\mathbf{I}}{dt} + \mathbf{R}\mathbf{I} + \mathbf{F}^T \Phi = -\frac{d\mathbf{V}_S}{dt}, \quad (9)$$

with  $\mathbf{I}(0) = \mathbf{I}_0$ , where  $\mathbf{I}_0$  is the prescribed initial condition.

In (9) the unknowns are

- $\mathbf{I}(t)$  is a column vector discretizing the unknown density current  $\mathbf{J}(\mathbf{r}, t)$  (i.e.,  $\mathbf{I}(t) = [I_i(t)]$ )
- $\Phi(t)$  is a column vector representing the unknown voltages at the electrodes (i.e.,  $\Phi(t) = [\Phi_h(t)]$ )

In the RHS of (9) appears the source term  $\mathbf{V}_S(t) = [V_{sk}(t)]$  given by

$$V_{sk}(t) = \int_{V_c} \nabla \times \mathbf{N}_k(\mathbf{r}) \cdot \mathbf{A}_s(\mathbf{r}, t) dV \tag{10}$$

The matrices in (9) are  $\mathbf{R} = [R_{ij}]$ ,  $\mathbf{L} = [L_{ij}]$ ,  $\mathbf{F} = [F_{ij}]$ . They are defined as

$$R_{ij} = \int_{V_c} \nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \sigma^{-1} \nabla \times \mathbf{N}_j(\mathbf{r}) dV, \tag{11}$$

$$L_{ij} = \frac{\mu_0}{4\pi} \int_{V_c} \int_{V_c} \frac{\nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \nabla \times \mathbf{N}_j(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dV dV', \tag{12}$$

$$F_{ij} = \int_{S_i} \nabla \times \mathbf{N}_j(\mathbf{r}) \cdot \hat{\mathbf{n}} dS. \tag{13}$$

The dimension of the matrix  $\mathbf{F}$  is  $N_E \times N$ . It is worth noticing that

- $F_{ij}$  is the contribution to the current flowing in the electrode  $i$  due to the DOF  $j$

The current  $C_i$  flowing in the  $i$ -th electrode, is the product of the  $i$ -th row ( $\mathbf{F}_i$ ) of  $\mathbf{F}$  times  $\mathbf{I}$ . That is:

$$C_i = \mathbf{F}_i \mathbf{I}. \tag{14}$$

Rewriting the condition (14) for each electrode we have

$$\mathbf{F} \mathbf{I} = \mathbf{I}_h, \tag{15}$$

- $F_{ij}$  is zero for those DOF  $j$ , which do not belong to the boundary.

Hence  $\mathbf{F}$  is sparse due to the local interaction of the electrodes.

Matrices  $\mathbf{R}$  and  $\mathbf{L}$  (whose dimensions are  $N \times N$ ) represent the discrete counterparts of the electric constitutive relationship (Ohm’s law) and of the vector potential operator, respectively. Matrices  $\mathbf{R}$  and  $\mathbf{L}$  are symmetric and positive definite. In addition, matrix  $\mathbf{R}$  is sparse, since it arises from a local operator, whereas matrix  $\mathbf{L}$  is fully populated, because it arises from the Biot–Savart integral operator.

Applying implicit time step integration to Equation (9), and keeping into account constrain (15) and the initial condition, the algebraical system to be solved is:

$$\begin{aligned} (\mathbf{L} + \Delta t \mathbf{R}) \mathbf{I}^{(k+1)} + \Delta t \mathbf{F}^T \Phi^{(k+1)} &= \mathbf{L} \mathbf{I}^{(k)} + \mathbf{V}_S^{(k+1)} - \mathbf{V}_S^{(k)} \\ \mathbf{F} \mathbf{I}^{(k+1)} &= \mathbf{I}_h^{(k+1)} \\ \mathbf{I}^{(0)} &= \mathbf{I}_0 \\ k &\geq 1 \end{aligned} \tag{16}$$

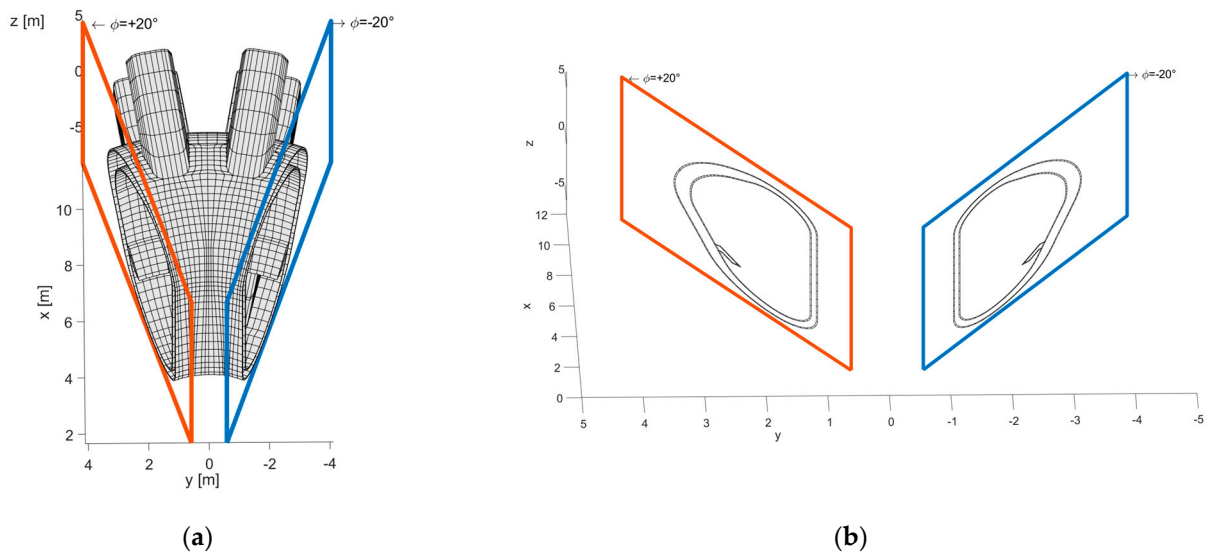
where

- $\Delta t$  is the time step
- $\mathbf{I}^{(k)} = \mathbf{I}(k\Delta t)$  and  $\Phi^{(k)} = \Phi(k\Delta t)$  are the unknowns at integration instant  $k\Delta t$
- $\mathbf{I}_h^{(k+1)} = \mathbf{I}_h((k+1)\Delta t)$  are the prescribed currents flowing in the electrodes at integration instant  $k\Delta t$
- and  $\mathbf{V}_S^{(k)} = \mathbf{V}_S(k\Delta t)$  are source terms at integration instant  $k\Delta t$

### 2.3. Symmetric Periodic Boundary Condition

In many relevant situations, two proper conditions are satisfied. First, the plasma discharge is axisymmetric. Therefore, the plasma discharge can also be retained periodically, with the same period of the nuclear fusion reactor. Second, the electrodes present on each sector and used to inject electrical energy into the system, may be driven by the same waveforms. Under these two conditions, the electromagnetic fields are periodic with the period of the nuclear fusion reactor. Similar arguments can be used in the presence of other symmetries, other than rotations.

For the sake of clarity, in Figure 2 we report a simple case in which a periodic boundary condition should be set at  $\varphi = -20^\circ$  and  $\varphi = +20^\circ$  on a symmetry mesh. We should force the density current  $\mathbf{J}(\mathbf{r}, t)$ , at each boundary point at  $\varphi = -20^\circ$ , to be equal to the correspondent point at  $\varphi = +20^\circ$ .



**Figure 2.** Forcing periodic boundary conditions. (a) The sector mesh ( $-20^\circ < \varphi < 20^\circ$ ); (b) The boundary faces at  $\varphi = -20^\circ$  and  $\varphi = +20^\circ$ . The current flowing at each face at  $\varphi = -20^\circ$  should be equal to the current flowing in the correspondent face at  $\varphi = +20^\circ$ .

Let  $\alpha$  be defined as the semi-angular width of the sector. For each pair of facets  $f_1$  and  $f_2$  lying, respectively, at  $\varphi = +\alpha$  and  $\varphi = -\alpha$  and coupled by the condition imposed by the periodicity, we force the current flowing through  $f_1$  to be equal to the one flowing through  $f_2$ . This can be easily done by the means of the matrix  $\mathbf{F}$ . Keeping into account definition (15) we should have

$$C_{f_1} = C_{f_2}, \tag{17}$$

and hence

$$(\mathbf{F}_{f_1} - \mathbf{F}_{f_2})\mathbf{I} = 0 \tag{18}$$

Therefore, the periodic boundary conditions act like the electrode's conditions, adding an equation such as (18) for each corresponding pair of facets.

Summing up, in the presence of both electrode currents and periodic boundary conditions, the second equation in (16) is generalized as:

$$\mathbf{B} \mathbf{I}^{(k+1)} = \mathbf{M}^{(k+1)}, \tag{19}$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{F}_E \\ \mathbf{F}_\alpha - \mathbf{F}_{-\alpha} \end{bmatrix}, \tag{20}$$

$$\mathbf{M}^{(k+1)} = \begin{bmatrix} \mathbf{I}_h^{(k+1)} \\ 0 \end{bmatrix} \tag{21}$$

$$\mathbf{F}_E \stackrel{\text{def}}{=} \{F_{ij} \mid i \in S_E, j \in 1, N\}$$

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \{F_{ij} \mid i \in \pi_\alpha, j \in 1, N\}$$

$$\mathbf{F}_{-\alpha} \stackrel{\text{def}}{=} \{F_{ij} \mid i \in \pi_{-\alpha}, j \in 1, N\}$$

where  $\pi_\alpha$  and  $\pi_{-\alpha}$  are the planes at  $\varphi = +\alpha$  and  $\varphi = -\alpha$ , respectively.

In accordance with the two different constrains, the matrix  $\mathbf{B}$  has a number of row equal to  $N_E + N_p$ , where

$N_E$  is the number of electrodes on  $S_E$  where the current should be prescribed.

$N_p$  is the number of faces at  $\varphi = +\alpha$  (or at  $\varphi = -\alpha$ ).

Finally, in place of the (16), the overall numerical system to be solved is:

$$\begin{aligned} (\mathbf{L} + \Delta t \mathbf{R})\mathbf{I}^{(k+1)} + \Delta t \mathbf{B}^T \boldsymbol{\Phi}^{(k+1)} &= \mathbf{L}\mathbf{I}^{(k)} + \mathbf{V}_S^{(k+1)} - \mathbf{V}_S^{(k)} \\ \mathbf{B} \mathbf{I}^{(k+1)} &= \mathbf{M}^{(k+1)} \\ \mathbf{I}^{(0)} &= \mathbf{I}_0 \\ k &\geq 1 \end{aligned} \tag{22}$$

At the discrete level, the dynamical matrix is:

$$\mathbf{A} = \mathbf{R} \Delta t + \mathbf{L}. \tag{23}$$

The solution of large-scale problems involving a fully populated matrix poses a relevant challenge both in the assembly and in the solution of the dynamical system (22) [26]. Indeed

- (i) The assembly of  $\mathbf{L}$  has a cost of  $O(N^2)$ .
- (ii) The inversion (usually factorization) of the matrix  $\mathbf{A}$ , using a direct method, as well know, requires  $O(N^3)$  operations.

The authors developed a parallel implementation of the direct method to solve system (22) [26,27], based on the Scalapack library [28].

### 3. QR-Recursive Method

#### 3.1. Summary of QR-Recursive Approach

As explained in the previous section, the solution of (22) cost is prohibitively large when the mesh of the structure is very detailed. The compression of the stiffness matrix is recommended only for large scale problems, otherwise a direct method is preferred. When working on large scale problems, the compression of such large matrices as a whole is too expensive, from the computational point of view. Therefore, we are obliged to resort to recursive approaches, such as the QR-recursive method proposed in the present work.

In iterative methods, the matrix-vector product is the fundamental building block of the solution, and the QR-recursive method [20,21] is actually a way to evaluate efficiently the matrix-vector product.

The direct computation of the matrix-vector product  $\mathbf{A}\mathbf{x}$  (see [29]) is actually too costly, from the numerical point of view (it costs  $O(N^2)$  operations); QR-based methods [30], compress the matrix and recast the product evaluation in a cheap cost (which scales  $O(N)$ ) by the means of separation between the near and far interactions. The near interactions should be computed without approximation; on the contrary, the far interaction could be approximated.

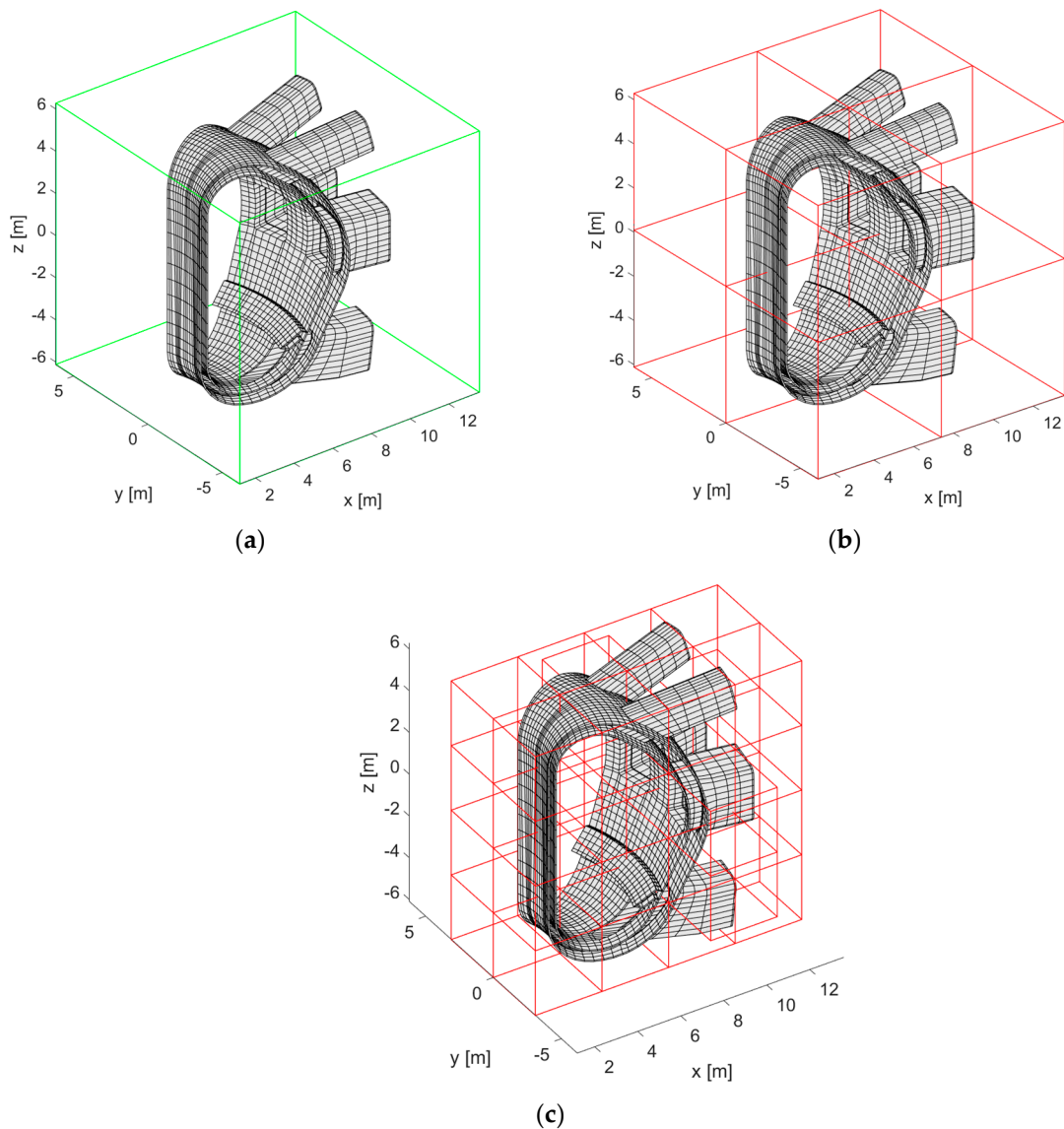


The relevant steps of the QR-recursive algorithm are:

- (1) Boxes generation
- (2) Definition of DOFs associated to the boxes
- (3) Definition of interaction matrix between two boxes
- (4) Setting  $\mathbf{L}_{\text{near}}$ ,  $\mathbf{L}_{\text{far}}$
- (5) Overall compressed operator optimization and practical considerations

### 3.1.1. Boxes Generation

The first step is grouping the mesh “objects” into clusters (**boxes**). The structure is recursively generated by subdividing the domain in cubic cells, halving the edge length at each subdivision. This procedure starts from the smallest cube containing the whole mesh (**b0**) and ends when a prescribed minimum number (*smi*n1) of “objects” is left in the box. The resulting boxes, which should not require further division, are called childless boxes. In the Appendix A, we report the full algorithm used, and in Figure 3 we show an example of an application.



**Figure 3.** Boxes Generation algorithm. (a) The **b0** at level = 0. (b) The eight boxes (children) generated at level equal 1 by subdivision of the box **b0**. (c) Shows the third level subdivision. Note that empty boxes are deleted.



Here, we use the elements as mesh “objects”. Hence, we refer to this method as the **ELEMENT-based** method.

As a final remark, we highlight that the shape of the elementary cell, the cube in this work, has to satisfy the following three properties:

- (a) The cell is able to tessellate the 3D space.
- (b) The cell can be exactly subdivided in eight smaller replicas.
- (c) The cell should present an aspect ratio of the order of unity. In other words, the cell should not be either flattened or elongated.

These conditions, bring naturally to cubic cells.

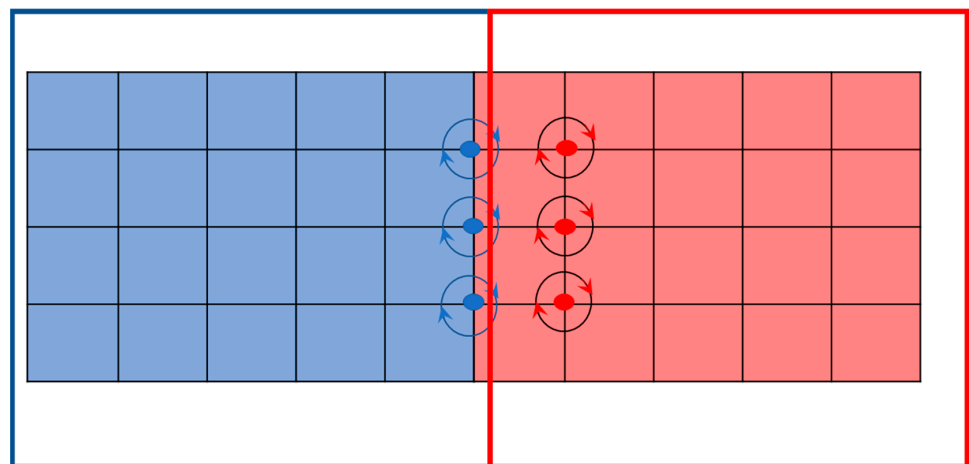
### 3.1.2. Definition of DOFs Associated to the Boxes

Once the boxes are created, we can define the DOFs to insert into the boxes. Let  $E$  and  $D$  be defined as

$E(\mathbf{b}) \stackrel{\text{def}}{=} \text{is the set of the all elements mesh belonging to the box } \mathbf{b}$

$D(\mathbf{b}) \stackrel{\text{def}}{=} \text{is the set of all DOFs } \in E(\mathbf{b})$  (24)

Note that the support of a given  $\text{DOF} \in D(\mathbf{b})$ , could not necessarily belong to  $E(\mathbf{b})$ . This actually occurs if the DOF lies on the boundary of the box. So, we label the DOFs as the internal DOF or boundary DOF (see Figure 4).



**Figure 4.** ELEMENT-based approach. Each element is associated to the box in which its barycenter lies. The two boxes element are marked blue and red, respectively. The blue DOFs are boundary DOF the red one are internal.

### 3.1.3. Definition of the Interaction Matrix between Two Boxes

Using the definition in (24), we are able to define the interactions matrix between a field  $\mathbf{b}$  and source box  $\mathbf{c}$ .

The matrix interaction, namely  $\mathbf{L}^{\mathbf{bc}}$ , is defined as

$$\mathbf{L}^{\mathbf{bc}} = \{ \mathbf{L}_{ij}^* \}, \quad i \in D(\mathbf{b}), \quad j \in D(\mathbf{c}), \tag{25}$$

where

$$\mathbf{L}_{ij}^* = \sum_{r \in E(\mathbf{b})} \sum_{s \in E(\mathbf{c})} \mathbf{L}_{i,j,r,s}. \tag{26}$$

The terms  $\mathbf{L}_{i,j,r,s}$  represent the element–element contribution to the computation of  $\mathbf{L}_{ij}$  (see (12)). It is easy to understand that

- $\mathbf{L}_{ij}^* \neq \mathbf{L}_{ij}$  for each matrix entry of  $\mathbf{L}^{\mathbf{bc}}$  for which  $i$  or  $j$  is a boundary DOF.

This is because some element–element interactions are missing.

- $\mathbf{L}_{ij}^* = \mathbf{L}_{ij}$  for each matrix entry of  $\mathbf{L}^{bc}$  for which both  $i$  and  $j$  are internal DOFs.

Hence, in the ELEMENT-based approach, the box–box interaction matrix  $\mathbf{L}^{bc}$  could not be a submatrix of the full matrix  $\mathbf{L}$  (see Figure 4).

Nevertheless, the  $\mathbf{L}$  matrix will be consistently reconstructed by superposition of all the  $\mathbf{L}^{bc}$  contributions.

### 3.1.4. Setting $\mathbf{L}_{near}, \mathbf{L}_{far}$

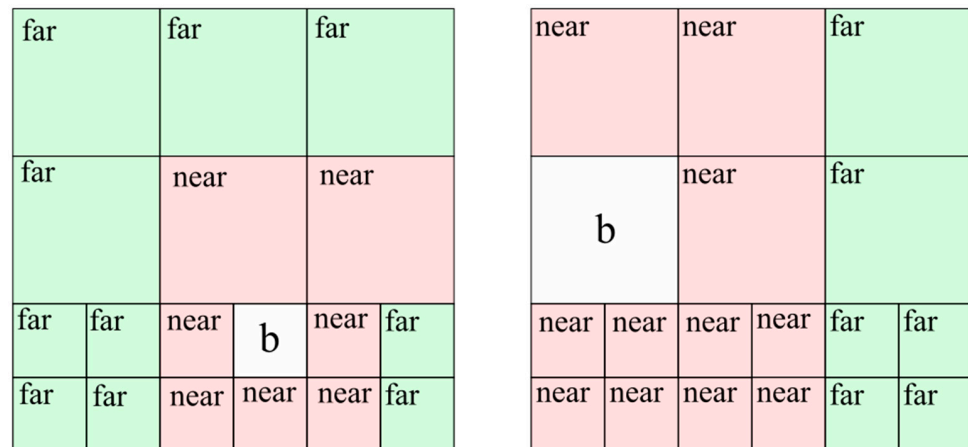
The last step is the identification of the near and far box–box interactions.

For each box–box interaction (namely  $\mathbf{b}, \mathbf{c}$ ) in which  $\mathbf{b}$  is field and  $\mathbf{c}$  is the source box, we define:

- $\mathcal{L}_{far}(\mathbf{b}) \stackrel{\text{def}}{=} \mathbf{c}$  is the set of sources boxes  $\mathbf{c}$ , such that the box  $\mathbf{b}$  is in far zone of the box  $\mathbf{c}$ .
- $\mathcal{L}_{near}(\mathbf{b}) \stackrel{\text{def}}{=} \mathbf{c}$  is the set of sources boxes  $\mathbf{c}$ , such that the box  $\mathbf{b}$  is in the near of the box  $\mathbf{c}$ .

The analytic definition of  $\mathcal{L}_{far}(\mathbf{b}), \mathcal{L}_{near}(\mathbf{b})$  is explained in Appendix B.

In Figure 5, we report the near and far zone generated by box  $\mathbf{b}$  acting as source.



**Figure 5.** The box  $\mathbf{b}$  is the source box. The near zone is depicted as pink, and the far zone is depicted as green.

Using such a box–box repartition, the  $\mathbf{L}$  matrix is written as follows:

$$\mathbf{L} = \mathbf{L}_{near} + \mathbf{L}_{far}. \tag{27}$$

$\mathbf{L}_{near}$  accounts for the box–box interactions, which should be computed exactly without approximation. There are two kinds of direct interactions:

- When the two boxes are classified as near, hence, they have a large rank and cannot be efficiently compressed.
- The interactions arising from non-local DOFs. These DOFs could not be compressed because they are related to nonlocal shape functions.

We call the  $\mathbf{L}_{add}$  matrix the sparse matrix arising from these non-local shape functions (see [31,32]).

Let  $N_{nbox}$  be defined as the number of the boxes, the  $\mathbf{L}_{near}$  is

$$\mathbf{L}_{near} = \mathbf{L}_{add} + \sum_{b=1}^{N_{nbox}} \sum_{c=1}^{N_{near}(\mathbf{b})} \mathbf{L}^{bc}. \tag{28}$$

$\mathbf{L}_{far}$  keeps into account the far distance box–box interactions. These kinds of interactions are low rank and, hence, they can be conveniently compressed. The compression

technique adopted is the modified Gram–Schmidt (MGS [22,33]). This algorithm returns an approximated QR-MGS factorization for each  $L^{bc}$ .

$$L^{bc} \cong Q^{bc}R^{bc} \tag{29}$$

Hence the matrix  $L_{far}$  is approximated as follows:

$$L_{far} = \sum_{b=1}^{N_{nbox}} \sum_{c=1}^{L_{far}(b)} L^{bc} \cong \sum_{b=1}^{N_{nbox}} \sum_{c=1}^{L_{far}(b)} Q^{bc}R^{bc}. \tag{30}$$

Doing this way, it is easy to show that both the computational cost and memory occupation are reduced. Indeed, let us consider two far boxes  $b$  and  $c$  having  $m$  and  $n$  objects, respectively. The matrix without compression (i.e.,  $L^{bc}$ ) has dimension  $m \times n$ , whereas in its compressed matrix  $Q^{bc}$  has dimension  $m \times r$ , and  $R^{bc}$  has dimension  $r \times n$ , being  $r$  the rank of the interaction. Please note that for far interactions

$$r \ll m, n. \tag{31}$$

Let  $x_c$  be the vector representing the sources in box  $c$ , the direct evaluation of the product  $L^{bc}x_c$  has a computational cost

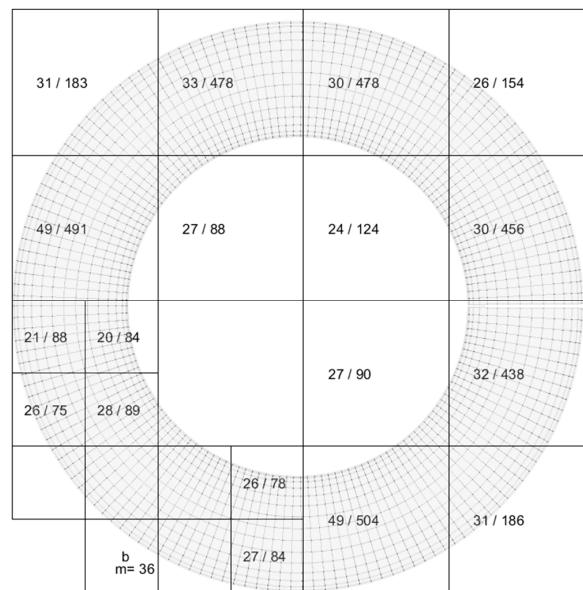
$$C_{dir} = m \times n, \tag{32}$$

whereas its approximated version  $Q^{bc}(R^{bc}x_c)$  costs

$$C_{app} = (m + n) \times r. \tag{33}$$

In force of the (31) it results in that for the far interactions  $C_{app} \ll C_{dir}$ . This is the heart of the compression method.

For sake of clarity, in Figure 6 we report the rank distribution of the matrix  $L^{bc}$ , where  $b$  is a given field box and  $c$  represents all the other source boxes. As expected, the rank reduces as distance increases.



**Figure 6.** The rank between the field box  $b$  and all the other source boxes (the  $c$  boxes). The number of the DOFs in the field box  $b$  is 36. For all the far sources boxes  $c$  it is reported in the format X/Y format, which stands for the rank (X) of the matrix  $L^{bc}$  and the number of sources DOFs (Y). Note that the X/Y information is omitted for the near boxes.

Moreover, we stress that the computational cost of  $\mathbf{L}^{bc}\mathbf{x}_c \cong \mathbf{Q}^{bc}\mathbf{R}^{bc}\mathbf{x}_c$  is exactly equal to the memory required to store the compressed version of the interaction. Hence, all the considerations done for the memory hold for the computational cost. This is a key feature of the QR-recursive method.

Using (28) and (30), the dynamic matrix appearing in (23), can be approximated as

$$\mathbf{A} \approx \mathbf{A}_{qr} = \mathbf{A}_{near} + \mathbf{A}_{far}, \quad (34)$$

where

$$\mathbf{A}_{far} = \mathbf{L}_{far}, \quad (35)$$

$$\mathbf{A}_{near} = \Delta t \mathbf{R}_{add} + \mathbf{L}_{add} + \sum_{b=1}^{N_{nbox}} \sum_{c=1}^{L_{near}(b)} \left( \mathbf{R}^{bc} \Delta t + \mathbf{L}^{bc} \right). \quad (36)$$

where  $\mathbf{R}_{add}$  is the matrix  $\mathbf{R}$  evaluated in the same sparsity pattern as for  $\mathbf{L}_{add}$ .

Moreover, note that the matrix  $\mathbf{R}$ , being a local sparse matrix, appears only in the near part.

We should remark that in implementation of CARIDDI code, in order to face huge cases, the parallelization is fully applied by the means of MPI (Message Passing Interface) library [34]. Indeed, the computation burden

- (i) assembling of  $\mathbf{L}_{near}$  and  $\mathbf{L}_{far}$
- (ii) product evaluation of  $\mathbf{A}_{qr}\mathbf{x}$  (i.e.,  $\mathbf{p}_{near} = \mathbf{L}_{near}\mathbf{x}$ ,  $\mathbf{p}_{far} = \mathbf{L}_{far}\mathbf{x}$ )

Is approximatively equally distributed among processes used in computation.

As a matter of fact, this means that performances of the QR method scale linearly versus the number of processes used in computation.

Indeed, we will show (see Section 3.2) that the computational cost of the QR-recursive method, is directly connected to the time required to evaluate the compressed product  $\mathbf{A}_{qr}\mathbf{x}$ . In turn, this time, as aforementioned, depends on the memory used. When this memory is equally redistributed among  $N_p$  processes, the computation cost is reduced by a factor one over  $N_p$ .

In the numerical sections, we will report some results about the numerical performances of the parallelization [28,35].

### 3.2. The Preconditioner and the Initial Guess Estimation

As said in previous paragraphs, the QR-recursive method uses an iterative solver for the solution of the algebraic system, which takes advantage from the compressed version of the matrix-vector product. In present work, the iterative solver adopted is GMRES [36]. As it is well known, the main operations required by an iterative solver are: (i) the computation of the matrix-by-vector product  $\mathbf{A}\mathbf{x}$  and, (ii) the preconditioning. The computational cost of an iterative solver, then, is proportional to the number of multiplications required for the product and to the number of iterations required to achieve convergence, assuming that the preconditioning has a cheaper evaluation. The role of the preconditioner is to reduce the number of iterations and, ultimately, the computational cost. In order to understand how this preconditioner works, we apply the left preconditioner to the first equation in (22). Assuming for sake of simplicity, the system is without electrodes, we have

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{I}^{(k)} = \mathbf{P}^{-1}\left(\mathbf{L}\mathbf{I}^{(k-1)} + \mathbf{V}_S^{(k+1)} - \mathbf{V}_S^{(k)}\right). \quad (37)$$

The role of the preconditioner  $\mathbf{P}$  is to reduce the condition number of the stiffness matrix  $\mathbf{P}^{-1}\mathbf{A}$ . The “ideal” preconditioner gets this number close to unity and it is a kind of inverse of  $\mathbf{A}$ . A critical issue in this framework is the dependence of matrix  $\mathbf{A}$  on  $\Delta t$ . In fast transient analysis,  $\Delta t$  has to be small enough to model properly the underlying dynamics. On the other hand, in slow transients  $\Delta t$  has to be large enough to reduce the computational cost. Therefore, the dependence of  $\mathbf{A}$  on  $\Delta t$  yields the dependence of  $\mathbf{P}$  on  $\Delta t$ . Specifically, for small  $\Delta t$  the preconditioner should be tailored on  $\mathbf{L}$ , whereas for large

$\Delta t$  the preconditioner should be tailored on  $\mathbf{R}$ , as follows from (23). In all remaining cases ( $\Delta t$  not too small nor too large) the preconditioner depends on  $\mathbf{R}$  and  $\mathbf{L}$ . The appropriate value of  $\Delta t$  depends on the underlying dynamics that are caused by the source, i.e., by the rate of change of  $\mathbf{V}_S(t)$  defined in (16) and appearing in (9).

It should be stressed that a preconditioner based on the  $\mathbf{L}$  matrix is still an open problem. From our knowledge and experience any attempt to insert a “light” modified version of  $\mathbf{L}$  in the preconditioner poorly fails.

In this work we set  $\mathbf{P} = \mathbf{R}$ . As we have said,  $\mathbf{R}$  is a sparse, symmetric, and positive definite matrix. Its factorization and application are numerically very effective [37]:

- The computational cost (memory and time) of the factorization is linear versus the number of unknowns. Actually, it uses Cholesky decomposition (tailored for sparse matrix).
- The back/forward substitution (involved in the Cholesky method) is also very cheap.

The  $\mathbf{R}$  preconditioner is not very effective starting from a null initial value, for the initial guess of the iterative method.

Extensive numerical experiments performed on typical meshes used in fusion devices, have shown that, if we use an estimation for the initial value of the iterative solver, the  $\mathbf{R}$  preconditioner works quite fine, at least when the time variations are slow. Degradation of the preconditioner will appear only in limited time slots when abrupt transitions occur. The simple time estimation scheme that we use for the initial value is

$$\mathbf{I}_0^{(k)} = 2\mathbf{I}^{(k-1)} - \mathbf{I}^{(k-2)}, \quad (38)$$

where  $\mathbf{I}_0^{(k)}$  is the initial guess for GMRES at the current instant  $k$  and  $\mathbf{I}^{(k-1)}$  and  $\mathbf{I}^{(k-2)}$  are the solutions at one and two previous time steps, respectively.

In the numerical results section, we report the GMRES number of iterations, proving the efficiency of the proposed strategy.

Note that, being very effective with the preconditioner, the performance of the iterative solver depends only by the computational cost of the approximated product  $\mathbf{A}_{qr}\mathbf{x}$ .

### 3.3. Extension of Compression to Symmetries

In the present work, we introduce a simple approach in order to apply compression whenever symmetries are present in the structure.

Hereafter we assume that (i) the nuclear fusion reactor is a periodic structure, (ii) the plasma discharge is axisymmetric, and (iii) the electrodes present on each sector are driven by the same waveform (see Section 2.3). Assumption (i) is typical in nuclear fusion machines. Assumptions (ii) and (iii) are not restrictive at all. Indeed, if not satisfied, the approach proposed in this paper can be extended to these cases by exploiting the framework of [38]. Hereafter, for the sake of simplicity, we present the compression in the presence of symmetries, under assumptions (i)–(iii).

To describe the approach, we briefly recall how they are handled in the CARIDDI code. Two classes of symmetries are handled. One is the symmetry of reflection with respect to a co-ordinate plane; the other one is the symmetry with respect to a given rotation around the axis  $z$ . The underlying idea, in order implement such a symmetry condition, is based on two considerations:

1. Reducing the solution domain  $V_c$  only to an elementary part of the whole structure. In the following, this part of the domain is called the main sector.
2. Assuming that basis functions  $\nabla \times \mathbf{N}_i(\mathbf{r})$  automatically verify the symmetry conditions, by two suitable operators: reflection and rotation.

For instance, with reference to a system of rectangular coordinates with unit vectors  $\hat{\mathbf{t}}_x$ ,  $\hat{\mathbf{t}}_y$ , and  $\hat{\mathbf{n}}_z$ , with  $\hat{\mathbf{n}}_z$  perpendicular to the plane of symmetry, we define the following reflection operator:

$$\mathbf{S}_n = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{vmatrix}, \quad (39)$$

such that the components of  $\mathbf{J}_k$  at the reflected point  $\mathbf{r}_r = \mathbf{S}_n \mathbf{r}$  are given by

$$\mathbf{J}_i(\mathbf{r}_r) = \varepsilon \mathbf{S}_n \mathbf{J}_i(\mathbf{r}), \quad (40)$$

where  $\mathbf{r} = x\hat{\mathbf{t}}_x + y\hat{\mathbf{t}}_y + z\hat{\mathbf{n}}_z$  is in the integration domain  $V_c$ ,  $\mathbf{r}_r$  is in the reflected domain, the 3D vectors are represented as column vectors made of their Cartesian components, and  $\varepsilon$  is +1 or -1, depending on the type of symmetry. It should be noticed that the continuity of  $\mathbf{J} \cdot \hat{\mathbf{n}}_z$  has to be assured on the symmetry planes. Therefore, for instance, when applying a reflection with  $\varepsilon = +1$ , the condition  $\mathbf{J} \times \hat{\mathbf{n}}_z = 0$  is automatically guaranteed on the symmetry plane.

Similarly, the rotation of an angle  $\alpha$  around the z axis can be represented by the usual operator of rotation  $\mathbf{R}_\alpha$ :

$$\mathbf{R}_\alpha = \begin{vmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{vmatrix}, \quad (41)$$

such that the components of  $\mathbf{J}_i$  at the rotated point  $\mathbf{r}_r = \mathbf{R}_\alpha \mathbf{r}$  are given by:

$$\mathbf{J}_i(\mathbf{r}_r) = \mathbf{R}_\alpha \mathbf{J}_i(\mathbf{r}). \quad (42)$$

Doing it in such a way, the solution in the main sector, along with the operators defined in (40) and (42), ensures the knowledge of the solution in the rest of the structure (that is in the whole 360° torus).

At this point, the calculation of the coefficients of  $\mathbf{L}$ ,  $\mathbf{R}$ , and  $\mathbf{V}$  is straightforward, and the evaluation of the matrices  $\mathbf{L}$ ,  $\mathbf{R}$ ,  $\mathbf{V}$  are reduced only to the DOFs in main sector. Indeed, using a number of symmetries equal to  $N_{\text{sym}}$  and a number of rotations equal to  $N_{\text{rot}}$ , they have the following form:

$$L_{ij} = 2N_\alpha \sum_{m=0}^{N_{\text{rot}}} \sum_{j=0}^{N_{\text{sym}}} \frac{\mu_0}{4\pi} \int_{V_c} \int_{V_v} \frac{\nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}_\alpha^m \varepsilon^j \mathbf{S}_n^j \nabla \times \mathbf{N}_j(\mathbf{r}')}{|\mathbf{r} - \mathbf{R}_\alpha^m \mathbf{S}_n^j \mathbf{r}'|} d\mathbf{r}' d\mathbf{r}, \quad (43)$$

$$R_{ij} = -2N_\alpha \int_{V_c} \nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \nabla \times \mathbf{N}_j(\mathbf{r}) d\mathbf{r}, \quad (44)$$

$$V_{ij} = -2N_\alpha \int_{V_c} \nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \frac{\partial}{\partial t} \mathbf{A}_s(\mathbf{r}, t) d\mathbf{r}, \quad (45)$$

where  $N_\alpha = \frac{2\pi}{\alpha}$ .

Limiting the computation to the main sector has two obvious advantages:

- (i) The number of DOFs is reduced by a factor of  $n_{\text{tot}} = N_{\text{sym}} \times N_{\text{rot}}$ . This gives a huge gain in matrix storing, factorizing, and inverting.
- (ii) The integration (43) can be seen as limiting the outer integral defined in (12) only to the main sector. Clearly, this reduces the matrix  $\mathbf{L}$  assembly time.

We can summarize the symmetry approach, saying that the matrix  $\mathbf{L}$  can be seen as the mutual inductance between DOFs. Without symmetry the field and source DOFs are the same. Using the symmetry approach, the field DOFs are only in the main sector, whereas the sources DOFs should be considered distributed all over the 360°, but they are



implemented (by the means of the operator (40) and (42)) using only the solution contained in the main sector. Expanding (43) we have

$$\mathbf{L} = \mathbf{L}^{(1)} + \mathbf{L}^{(2)} + \dots + \mathbf{L}^{(n_{tot})}, \quad (46)$$

where  $\mathbf{L}^{(i-th)}$  can be seen as the effect on main sector DOFs due to the DOFs contained in the  $i - th$  source rotated sector.

Now that we have explained how symmetry/rotation are implemented in CARIDDI code, we are ready to face the symmetry/rotation regarding the compression of the matrix  $\mathbf{L}$ .

Let  $\mathbf{L}^{bc}$  be the interaction matrix between two far boxes,  $\mathbf{b}$  (field box) and  $\mathbf{c}$  (source box). We denote with  $m$  and  $n$  the number of DOFs in the field and source box, respectively. In order to evaluate  $\mathbf{L}^{bc}$ , as said before, we need to consider the sources boxes  $\mathbf{c}$  for all rotations and symmetry sectors. The resulting matrix will be obtained as superposition of

$$\mathbf{L}^{bc} = \mathbf{L}^{bc(1)} + \mathbf{L}^{bc(2)} + \dots + \mathbf{L}^{bc(n_{tot})}, \quad (47)$$

where  $\mathbf{L}^{bc(j)}$  with  $j = 1$  to  $n_{tot}$  represents the partial contribute of each sector/symmetry.

We have now two ways to apply the compression to  $\mathbf{L}^{bc}$ :

- (1) Compressing each single interaction matrix  $\mathbf{L}^{bc(j)}$  appearing in (47).
- (2) Summing the interactions matrices  $\mathbf{L}^{bc(j)}$  and after compressing the resulting matrix  $\mathbf{L}^{bc}$ .

In the first approach, being each term  $\mathbf{L}^{bc(j)}$  a low rank matrix, we can be applying to it the QR-MGS factorization. So, we can approximate equation (47) as:

$$\mathbf{L}^{bc} \approx \sum_{j=1}^{n_{tot}} \mathbf{Q}^{bc(j)} \mathbf{R}^{bc(j)}, \quad (48)$$

where  $\mathbf{Q}^{bc(j)}$  and  $\mathbf{R}^{bc(j)}$  have dimensions  $m \times r_j$ , and  $r_j \times n$ , respectively.

Keeping into account the (29), the computational cost (and memory) of the product  $\mathbf{Q}^{bc(j)} \mathbf{R}^{bc(j)} \mathbf{x}$ , is  $(m + n) \times r_j$ . Hence, the overall cost of the product  $\mathbf{y} = \mathbf{L}^{bc} \mathbf{x}$  is equal to:

$$c1 = \sum_{j=1}^{n_{tot}} (m + n) \times r_j = (m + n) \times n_{tot} \times r_m, \quad (49)$$

where  $r_m = \frac{1}{n_{tot}} \sum_{j=1}^{n_{tot}} r_j$  is the medium rank.

The second approach is much simpler. The low rank nature of each  $\mathbf{L}^{bc(j)}$  implies that the overall matrix summation  $\mathbf{L}^{bc}$  is low rank too. Although, it must be said that its rank generally could be greater than the  $\mathbf{L}^{bc(j)}$  rank in the sum. Hence, in the second proposed approach, we evaluate directly the QR-MGS factorization:

$$\mathbf{L}^{bc} \approx \mathbf{QR}. \quad (50)$$

Its computational cost is:

$$c2 = (m + n) \times r_{all}. \quad (51)$$

In order to compare the two approaches, we simply compute the ratio between (49) and (51) and we have

$$\frac{c1}{c2} = \frac{n_{tot} \times r_m}{r_{all}} \quad (52)$$

Extensive numerical experiments carried out on typical cases used in practical fusion devices, show that  $r_m \approx r_{all}$ . So as a matter of fact,  $c2/c1$  results to be of the order of  $n_{tot}$ , and hence, the second approach is to be preferred.